# MODEL-BASED DESIGN OF RESILIENT SYSTEMS USING QUANTITATIVE RISK ASSESSMENT

*B. L. Mediouni[1], Iulia Dragomir[2], A. Nouri[1], S. Bensalem[1,3]*

*[1] Huawei Technologies France S.A.S.U*

*[2] GMV Aerospace and Defence, Spain*

*[3] University Grenoble Alps, France*

VECoS 2020

This work has been supported by the European Union's Horizon 2020 research and innovation programme under grant agreements #730080 (ESROCOS) and #700665 (CITADEL)

# OUTLINE

INTRODUCTION

RESILIENT SYSTEMS DESIGN APPROACH

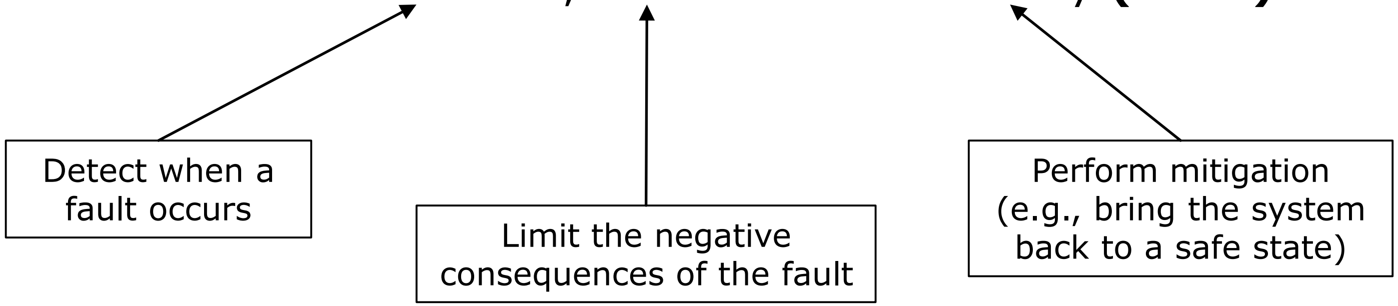
CASE STUDY

CONCLUSION

# INTRODUCTION

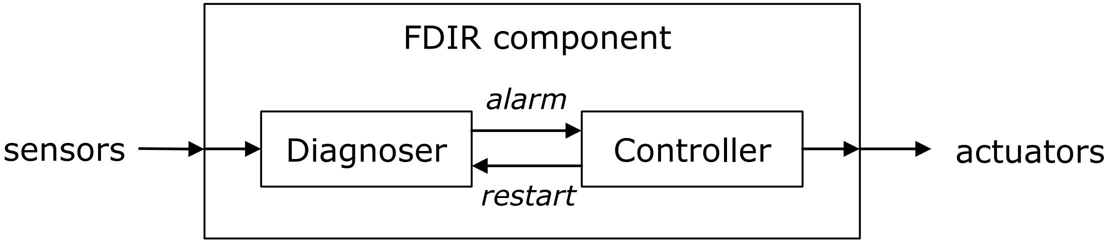
# RESILIENCE IN REAL-TIME SYSTEMS

- Mission and safety critical systems must satisfy a plethora of RAMS requirements
  - Harder to guarantee when they operate in uncertain environments
- Continuous safe behavior should be guarantee at execution time:
  - Need for **F**ault **D**etection, **I**solation and **R**ecovery **(FDIR)**



# FDIR

- An FDIR component consist of two main parts (functionalities):
  - The fault detection (i.e., diagnoser)
  - The fault recovery (i.e., controller)



# FDIR DESIGN

- Approach proposed by Dragomir at al. for real-time systems
  - One diagnoser per detectable fault
    - A diagnoser is built for each of the system’s faults
    - Not all faults have an impact on the system requirements
    - The system performance can be greatly degraded due to the large number of unnecessary components
  - Manually modeled controllers implementing recovery strategies
    - Controllers need to be verified
    - The validation problem is hard and possibly unfeasible due to model-checking scalability

# PROPOSED IMPROVEMENTS

- Quantify the impact of faults on the system requirements: **quantitative risk assessment**
  - Identify only those relevant faults (including combinations) for the system
  - Prioritize the faults for which diagnosers should be built
- Apply scalable techniques to validate the manually designed controller
  - Use **statistical model-checking** to gain confidence on the system’s correctness

# CONTRIBUTION

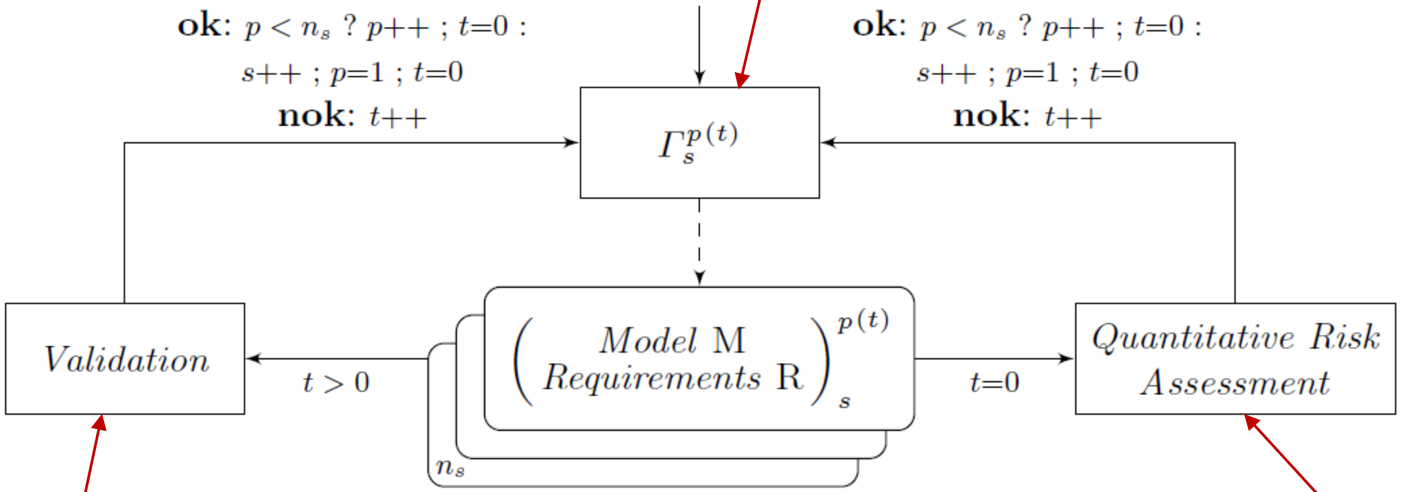
- A methodology for the design of resilient systems (i.e., with FDIR capabilities)
  - Iterative and incremental design process
  - Spiral development process with quantitative risk assessment and system validation
  - Partial automation with statistical model checking (SMC)
- Application on a real-life robotics case study
  - Inspired from planetary exploration missions
  - Three systems designs at different levels of granularity from general system design to deployment

# RESILIENT SYSTEMS DESIGN APPROACH

# METHODOLOGY

Model transformation where:

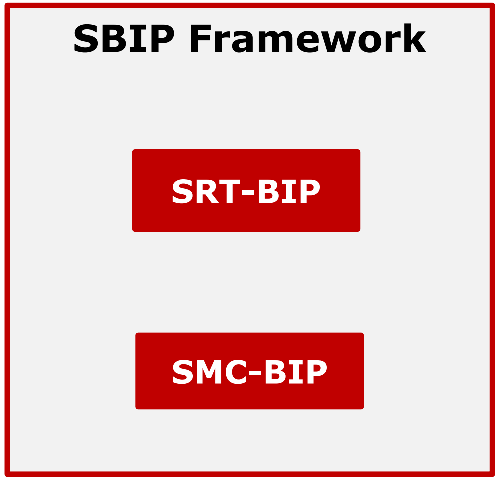
- Index s: refinements towards concrete implementation
- Index p: alternative configurations
- Index t (>0): model correction



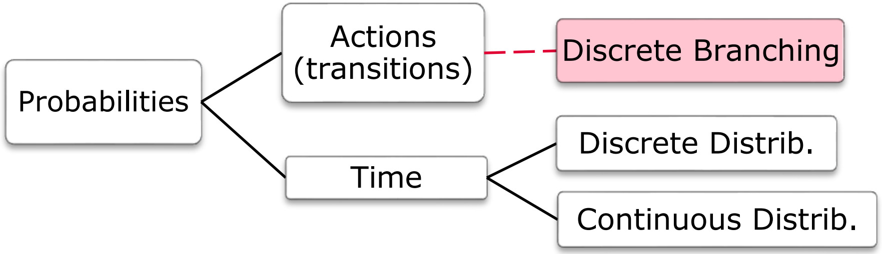

- Ensure the efficiency of the transformation
- Based on SMC

- Measure the impact of faults on requirements
- Based on SMC

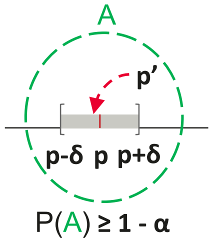
# SBIP FRAMEWORK



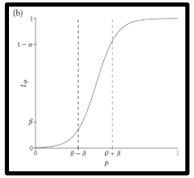

- **S**tochastic **R**eal-**T**ime **B**ehaviour **I**nteraction **P**riority
  - Design complex models exhibiting probabilistic behaviour on time and actions




- **S**tatistical **M**odel-**C**hecker:
  - **Quantitative analysis** (Probability Estimation):What is the probability that the system model M satisfies the property (requirement) φ?



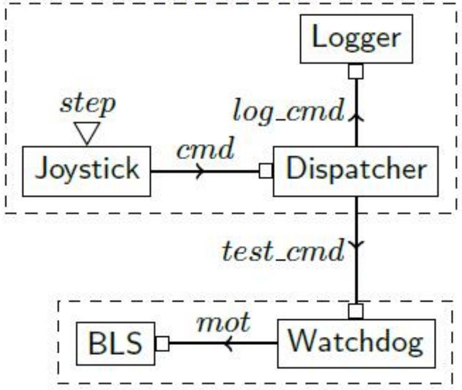

  - **Qualitative analysis** (Hypothesis Testing): Is the probability that the system M satisfies the property φ greater or equal than a threshold θ?



# CASE STUDY

# PLANETARY ROBOTICS

- Tele-operate a rover running SW developed with the ESROCOS toolchain (https://github.com/ESROCOS)
  - Drive remotely the rover with a joystick to different poses and acquire images
- Validate the toolchain and reusable components




(a) Architecture

(b) Bridget Rover (courtesy of Airbus)

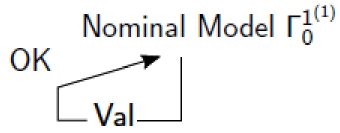
# SYSTEM SPECIFICATION

- **High-level design**:
  - 5 software components
  - Asynchronous communication
  - Periodic behavior
- **Formal model**:
  - 10 SRT-BIP components with a minimum of 4 variables each (clocks included)
  - External complex robotics data types
- **System requirements**:
  - All the commands sent are received by the locomotion software (no loss)
  - The locomotion software receives requests periodically (e.g., every 100ms)
- **Potential risks**:
  - Software-related:
    - The joystick fails to send periodic commands
  - Hardware-related:
    - The dispatcher looses the requests

# STEP 0: REQUIREMENTS

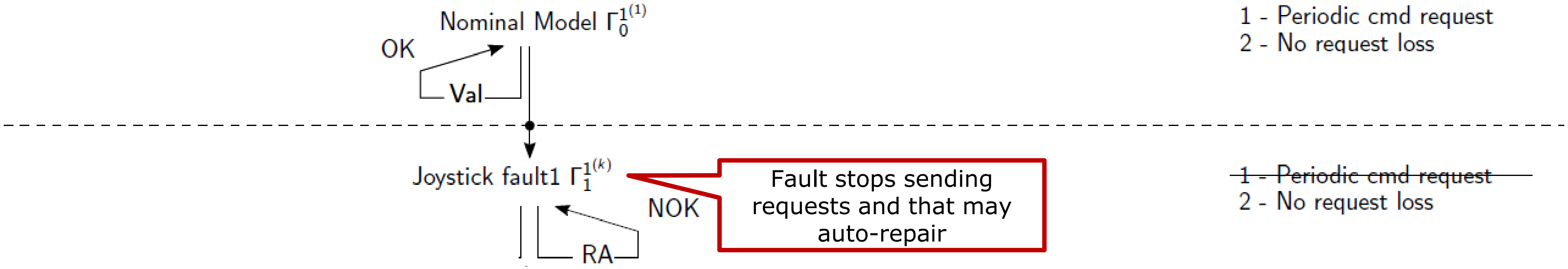
| ID | Formal specification |
|---|---|
| Requirements on the nominal system | |
| $\phi_0$ | $\square_{[0,10000]}\ (is\_sent \Rightarrow \lozenge_{[0,100]}\ is\_received\_c)$ |
| $\phi_1$ | $\square_{[0,10000]}\ (is\_received\_c \Rightarrow \lozenge_{[1,100]}\ is\_received\_c)$ |

# STEP 0: NOMINAL MODEL



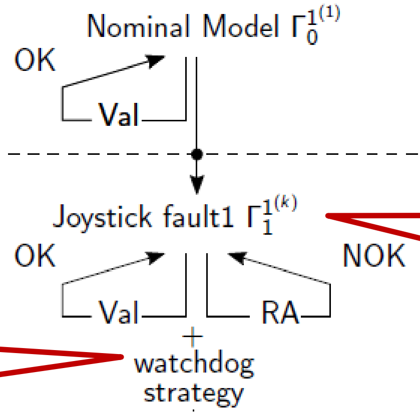

1 - Periodic cmd request
2 - No request loss

# STEP 1.0: MODEL WITH JOYSTICK FAULTS



CASE STUDY

# STEP 1.1: ADDING FDIR




1 - Periodic cmd request
2 - No request loss

Fault stops sending requests, joystick may auto-repair

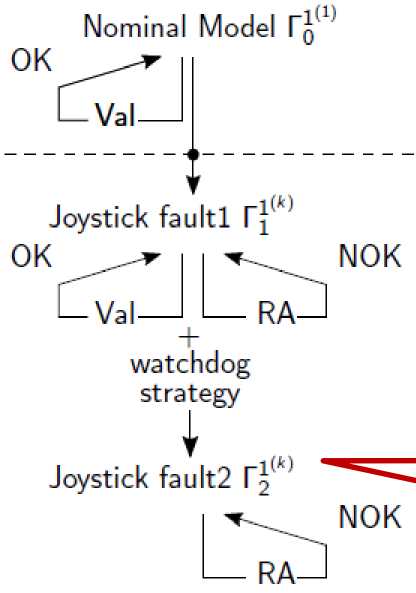
~~1 - Periodic cmd request~~
2 - No request loss

Trigger a timeout if nothing received after 110 ms (100 ms + some epsilon)

# STEP 1.1: REQUIREMENTS

| ID | Formal specification |
| --- | --- |
| **Requirements on the nominal system** | |
| $\phi_0$ | $\square_{[0,10000]} (is\_sent \Rightarrow \lozenge_{[0,100]} is\_received\_c)$ |
| $\phi_1$ | $\square_{[0,10000]} (is\_received\_c \Rightarrow \lozenge_{[1,100]} is\_received\_c)$ |
| **Requirements on the FDIR behavior** | |
| $\phi_2$ | $\square_{[0,10000]} (is\_sent \Rightarrow \lozenge_{[0,110]} is\_received)$ |
| $\phi_3$ | $\square_{[0,10000]} (is\_received \Rightarrow (\lozenge_{[1,110]} is\_received) \vee (\lozenge_{[110,200]} is\_timeout))$ |
| $\phi_4$ | $\square_{[0,10000]} (\lozenge_{[0,200]} nb\_received = nb\_sent + nb\_timeout)$ |

# STEP 2.0: PERFORMANCE-RELATED MODEL




1 - Periodic cmd request
2 - No request loss

~~1 - Periodic cmd request~~
2 - No request loss

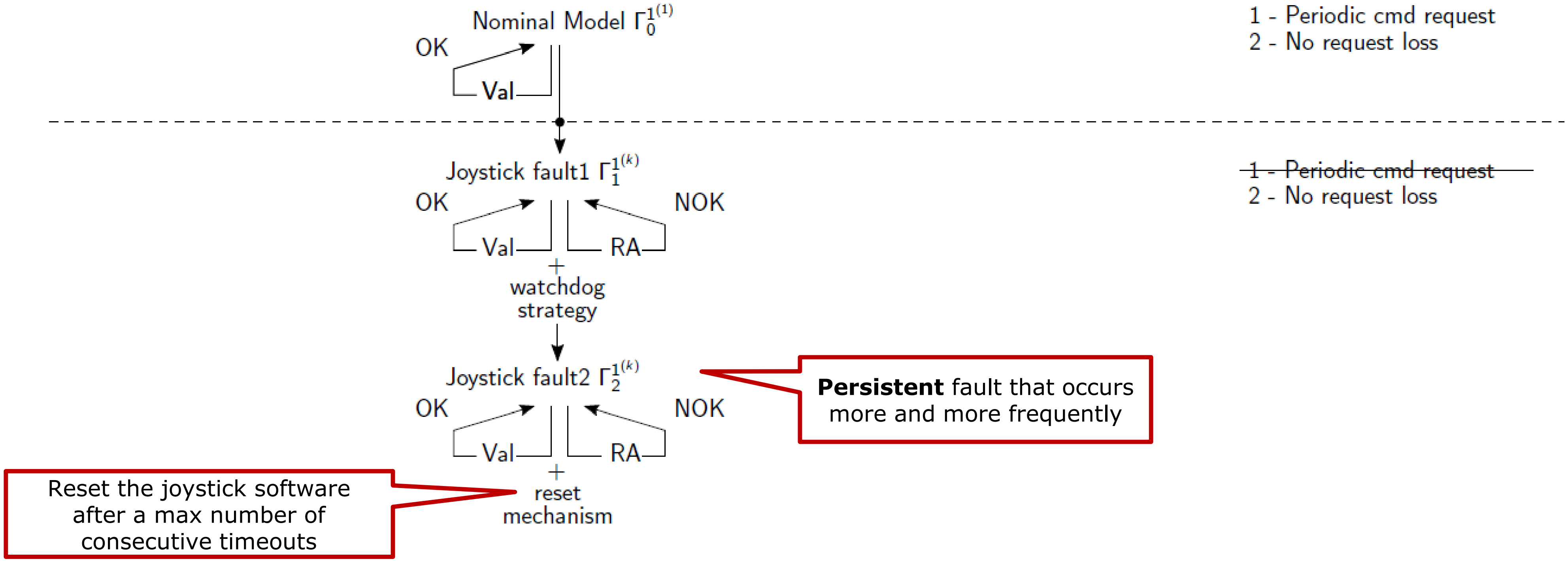
**Persistent** fault that occurs more and more frequently

CASE STUDY

# STEP 2.0: REQUIREMENTS

| ID | Formal specification |
|---|---|
| **Requirements on the nominal system** | |
| $\phi_0$ | $\square_{[0,10000]}\ (is\_sent \Rightarrow \lozenge_{[0,100]}\ is\_received\_c)$ |
| $\phi_1$ | $\square_{[0,10000]}\ (is\_received\_c \Rightarrow \lozenge_{[1,100]}\ is\_received\_c)$ |
| **Requirements on the FDIR behavior** | |
| $\phi_2$ | $\square_{[0,10000]}\ (is\_sent \Rightarrow \lozenge_{[0,110]}\ is\_received)$ |
| $\phi_3$ | $\square_{[0,10000]}\ (is\_received \Rightarrow (\lozenge_{[1,110]}\ is\_received) \vee (\lozenge_{[110,200]}\ is\_timeout))$ |
| $\phi_4$ | $\square_{[0,10000]}\ (\lozenge_{[0,200]}\ nb\_received = nb\_sent + nb\_timeout)$ |
| **Requirements on the system performance** | |
| $\phi_7(n)$ | $\square_{[0,10000]}\ (\lozenge_{[0,200]}\ nb\_timeout - (nb\_received - nb\_sent) \leq n)$ |
| $\phi_8(n)$ | $\square_{[0,10000]}\ (nb\_timeout \leq n)$ |
| $\phi_9(n)$ | $\square_{[0,10000]}\ (cnbt \leq n)$ |

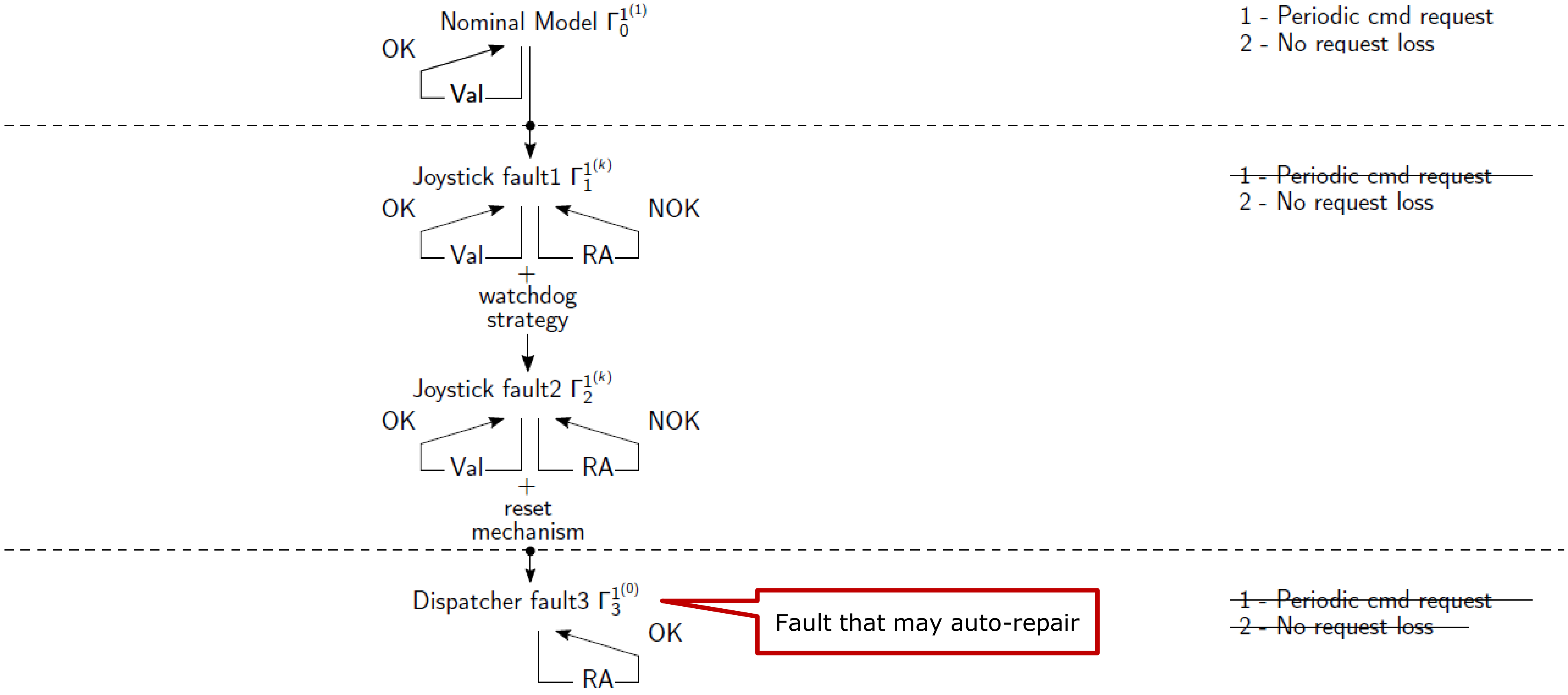
# STEP 2.1: ADDING RESET



# STEP 2.1: REQUIREMENTS

| ID | Formal specification |
|---|---|
| **Requirements on the nominal system** | |
| $\phi_0$ | $\Box_{[0,10000]}\ (is\_sent \Rightarrow \Diamond_{[0,100]}\ is\_received\_c)$ |
| $\phi_1$ | $\Box_{[0,10000]}\ (is\_received\_c \Rightarrow \Diamond_{[1,100]}\ is\_received\_c)$ |
| **Requirements on the FDIR behavior** | |
| $\phi_2$ | $\Box_{[0,10000]}\ (is\_sent \Rightarrow \Diamond_{[0,110]}\ is\_received)$ |
| $\phi_3$ | $\Box_{[0,10000]}\ (is\_received \Rightarrow (\Diamond_{[1,110]}\ is\_received) \vee (\Diamond_{[110,200]}\ is\_timeout))$ |
| $\phi_4$ | $\Box_{[0,10000]}\ (\Diamond_{[0,200]}\ nb\_received = nb\_sent + nb\_timeout)$ |
| $\phi_5$ | $\Box_{[0,10000]}\ (cnbt \geq MNBT \Rightarrow \Diamond_{[0,200]}\ is\_reset)$ |
| $\phi_6$ | $\Box_{[0,10000]}\ (is\_reset \Rightarrow \Diamond_{[0,100]}\ is\_received)$ |
| **Requirements on the system performance** | |
| $\phi_7(n)$ | $\Box_{[0,10000]}\ (\Diamond_{[0,200]}\ nb\_timeout - (nb\_received - nb\_sent) \leq n)$ |
| $\phi_8(n)$ | $\Box_{[0,10000]}\ (nb\_timeout \leq n)$ |
| $\phi_9(n)$ | $\Box_{[0,10000]}\ (cnbt \leq n)$ |

# STEP 3: MODEL WITH DISPATCHER FAULT



# CONCLUSION

# DISCUSSION (1/2)

- **Approach**:
  - ✓ **Flexibility** and rapid **exploration** of various situations thanks to the use of model-based approach
  - ✓ More **confidence** in the obtained results brought by formal methods
  - ✓ **Automation** of quantitative risk analysis and **scalability** provided by SMC
  - ✗ However, both the **identification** and the **evaluation** of risks remain manual and subject to the designer's interpretation
- **Tool**:
  - ✓ **Automation** of risk analysis enables design space exploration
  - ✓ Given a model and a property, **SMC** analysis is almost straightforward
  - ✗ Correct **formalization** of requirements in Metric Temporal Logic is required
  - ✗ Model **instrumentation** is needed to enable SMC analysis

# DISCUSSION (2/2)

- **Case study**:
  - ✓ Applied to a **real-life** robotics case study
  - ✓ FDIR component described here was **deployed** on the rover and **tested** during field trials
  - ✗ Identification of appropriate **abstraction** level and **probability distributions** require a deep knowledge of the system under analysis
  - ✗ **Wide** notion of risk requiring the analysis of risks at different levels, such as risks due to faults, risks due to adding new FDIR behavior, etc.
  - ✗ Managing the transformed **models** and the associated **requirements** can quickly become cumbersome

# FUTURE WORK

- Apply **qualitative assessment** in a first phase to exclude irrelevant risks
- Include knowledge-based techniques such as **machine-learning** for risk identification
- Evaluate the applicability of the proposed approach to security risk assessment

www.gmv.com

# THANK YOU

BLOG